20 Petaflops Simulation of Proteins Suspensions in Crowding Conditions

Massimo Bernaschi
Istituto Applicazioni Calcolo (CNR-IAC)
Consiglio Nazionale delle Ricerche
Rome, Italy
m.bernaschi@iac.cnr.it

Mauro Bisson
Istituto Processi Chimico-Fisici (CNR-IPCF)
Consiglio Nazionale delle Ricerche
Rome, Italy
bisson@iac.cnr.it

Massimiliano Fatica
Nvidia Corporation
Santa Clara, CA, USA
mfatica@nvidia.com

Simone Melchionna
Istituto Processi Chimico-Fisici (CNR-IPCF)
Consiglio Nazionale delle Ricerche
simone.melchionna@roma1.infn.it

ABSTRACT

We present performance results for the simulation of proteins suspensions in crowding conditions obtained with *MUPHY*, a computational platform for multi-scale simulations of real-life biofluidic problems. Previous versions of *MUPHY* have been used in the past for the simulation of blood flow through the human coronary arteries and DNA translocation across nanopores. The simulation exhibits excellent scalability up to 18,000 K20X Nvidia GPUs and achieves almost 20 Petaflops of **aggregate sustained performance** with a peak performance of 27.5 Petaflops for the most intensive computing component. Those figures demonstrate once again the flexibility of *MUPHY* in simulating biofluidic phenomena, exploiting at their best the features of the architecture in use. Preliminary results were obtained in the present case on a completely different platform, the IBM Blue Gene/Q. The combination of novel mathematical models, computational algorithms, hardware technology, code tuning and parallelization techniques required to achieve these results are presented.

1. INTRODUCTION

The behavior of proteins in crowding conditions is paramount to the regulation of cellular function and biological activity. Proteins are the main actors of cellular compartments having non-spherical shape and specific interactions among themselves and other ligands, with a macromolecular volume fraction reaching up to $\approx 40\%$ [1, 2, 3, 4]. Therefore, understanding cellular function requires a systematic analysis of how protein encounters take place, with diffusion being the main transport mechanism at low Reynolds numbers.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. SC '13, November 17 - 21 2013, , USA Copyright 2013 ACM 978-1-4503-2378-9/13/11$15.00.
http://dx.doi.org/10.1145/2503210.2504563

In the study of biological macromolecules by conventional computational methods, a limiting factor is the time-scale gap between experimental and computational trajectories. A classical all-atom Molecular Dynamics (MD) simulation of the cellular environment composed by several thousands proteins is currently beyond reach. This is due to the order of 10^4 degrees of freedom per protein involved, and principally, due to the inherent slow time scales related to dynamics and structural transformations, taking place over $\gg 10^6$ elementary timesteps.

Therefore this technique is too expensive to extract significant data related to long-range protein modifications and motion, due to the explicit representation of the solvent and the calculation of electrostatic interactions.

A multiscale computational approach is essential to study the structural and dynamical features of the macromolecular suspension. The solvent time scale is much shorter than the diffusive time scale of the dissolved proteins; interparticle interactions inducing additional time scales and hydrodynamic interactions affecting the short-time diffusive dynamics. Finally, at longer times, self-diffusion is affected by both hydrodynamic and direct interactions.

The rapid development of "force fields" allows studying proteins by substituting the all-atoms representation with a coarse-grained treatment. This class of force fields is experiencing burgeoning progress due to the performances in simulating proteins at an acceptable accuracy [5, 6]. The boost in performance becomes unacceptable when studying the suspension dynamics due to the neglect of the solvent and its structural and dynamical correlations. Fortunately, solvent-mediated interactions can be accounted for in the framework of the Brownian Dynamics technique, producing effective solvent-mediated interactions. Such choice represents a formidable algebraic challenge, as it involves computing a correlated thermal noise. This task requires the LU de-

composition of the hydrodynamic matrix, having $\sim 10^6 \times K^2$ elements, where K is the number of proteins in solution.

In this work, we employ multiscale computing to simulate a suspension made of thousands of proteins in presence of hydrodynamic interactions. The method uses a fully time-explicit simulation technique that offers advantages for the study of biofluids under realistic conditions. In particular, it enables scale-specific hydrodynamic interactions at sub-mesh spacing resolution, a crucial advantage to boost the performance of the simulations while guaranteeing numerical stability.

In order to resolve the fluid dynamics from the atomic to the global scale, we treat separately the solvent-mediated intramolecular and intermolecular hydrodynamic modes. The intramolecular hydrodynamic interactions belong to a fine scale, mediated by an instantaneous propagation of vorticity of the fluid solvent. The intermolecular modes instead, are of diffusional and viscous origin and belong to a manifold with much longer associated timescale than the intramolecular one.

The multiscale framework allows optimizing the number of computational resources required to study several proteins in solution. While the cost of including the hydrodynamic interactions scales with the cube of the total number of degrees of freedom, with our strategy the cost of the intramolecular interactions scales linearly with the number of proteins and with the cube of the number of particles per protein. Handling explicitly the fluid pattern on the coarse scale is again linear in the number of proteins involved.

For the simulations we used *MUPHY* (MUlti PHYsics simulator) [7]. *MUPHY* is a multiscale-multiphysics parallel code that we have developed, ported and optimized for several platforms, including clusters of GPUs.

The rest of the paper is organized as follows: Section 2 describes how multiscale computing can be applied to the study of proteins in crowding conditions. Section 3 describes the main features of the *MUPHY* code. Section 4 reports the main performance results. Section 5 reports physical results about protein diffuisivity. Finally, Section 6 summarizes the main findings and indicates future extensions of the present work.

2. MULTISCALE COMPUTING

Living systems are ubiquitously made of two kinds of components: a liquid, given by an aqueous solution, plasma, cytosol, etc., and suspended biomolecules, such as proteins, DNA, lipids and so forth. The dual nature of the generic biosystem, together with the wide diversity of solutes, applies to both the mesoscopic and the macroscopic scales of living systems. The *MUPHY* simulation platform handles the numerical solution of the fluid motion and suspended macromolecules, representing a major design effort in biofluidics research. Several solutions are available in order to model multiple types of biosystems. A non-exhaustive list of functionalities includes:

- selection of solutes as single particles, polymers and macromolecules;
- selection of Newtonian and non-Newtonian fluids, to reproduce from molecular hydrodynamics (Boltzmann kernel) [8] to stochastic frictional dynamics (Fokker-Planck kernel) [9];
- anisotropic [10] and scale-adaptive [11] particle-fluid coupling mechanisms;
- inclusion of stochastic fluctuations for the fluid and suspended particles;
- handling of generic confining environments [12];
- solution of electrostatics in presence of charged fluids [13];
- employing concurrent or staggered timestepping schemes for updating the fluid and particle states;
- regular and irregular domain partitionings for parallelism [14].

The *MUPHY* platform leverages two computational engines. The first one handles the motion of the generic fluid within the hydro-kinetic formulation embodied by the Lattice Boltzmann (LB) method [15]. In the hydro-kinetic approach, the motion of fluids in the continuum is reproduced by using collision-driven mechanisms (in contrast to the direct macroscopic description of fluid motion). The second engine handles the motion of Lagrangian particles by using a technique that borrows several technical aspects from MD. The nature of the suspended particles is non-conventional and requires a substantial extension of the basic MD technique.

In the present work we consider a concentrated suspension of macromolecules and label the collective positions and velocities of the macromolecular particles $\mathbf{R}$ and $\mathbf{V}$, respectively. The suspension is made of K identical macromolecules, each one with its own set of positions and velocities $\mathbf{R}_m, \mathbf{V}_m$, with $m = 1, ..., K$, being three-dimensional vectors with as many entries as the number of particles per macromolecule N_m.

At first glance, the motion of macromolecules in solution can be described by Langevin dynamics, such that each particle experiences a frictional force and a stochastic white noise stemming from the environment. The accuracy of the Langevin approach depends on the ratio between the solvent and the protein density, with the protein density being only 50% larger than the solvent. Because there is no clear separation of timescales between intramolecular and atom-solvent interactions, a generalized Langevin dynamics with memory should be used.

We account for such generalization by explicitly solving the Navier-Stokes evolution for the solvent. The momentum locally exchanged between the protein particles and the solvent generates hydrodynamic correlations. The coupling between fluid and particles takes place via specifically designed kernels based on kinetic modeling, significantly distinct from stresslet, boundary-integral and other methods based on macroscopic hydrodynamics. The solvent and particle dynamics schemes are tightly coupled and require the exchange of several quantities carrying the *fluid to particle* and *particle to fluid* interactions, as sketched in Fig. 1.

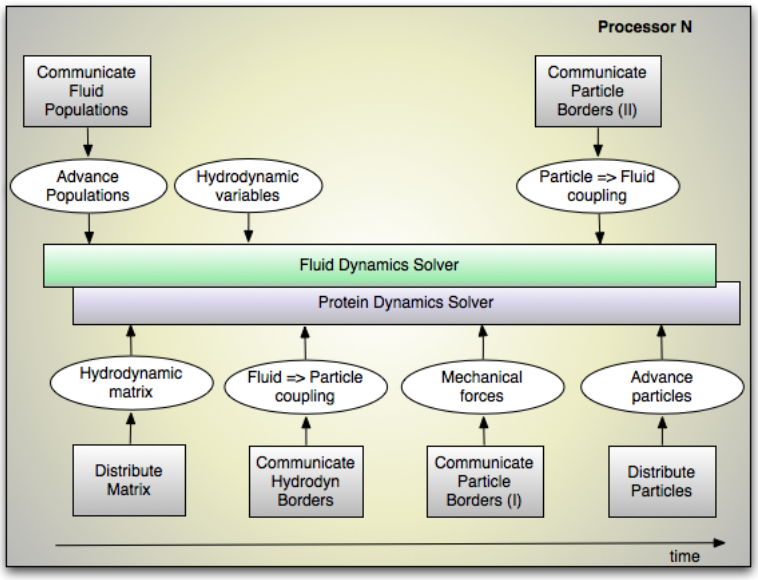

Figure 1: Coupling between the various *MUPHY* components on a given processor. The upper section takes care of the evolution of the solvent explicit dynamics (LB engine), the calculation of hydrodynamic variables and the exchange of particles momentum on the fluid. The lower section evolves the proteins by: i) computing the hydrodynamic matrix and performing the Brownian dynamics step, ii) computing the protein intra and inter-molecular forces, iii) computing the effect of the explicit fluid on the particles and iv) evolving the particles according to the explicit solvent. Each task is accompanied by a corresponding communication step to exchange fluid or particles over the domain borders. The communication of particles laying at the domain borders is needed for the calculation of mechanical and hydrodynamical forces and requires two different calls, due to the intrinsically different cutoffs involved in the two tasks.

During their evolution, the macromolecular movements are induced by both mechanical and hydrodynamic interactions. The former are due to excluded volume, electrostatic, bonding and hydrogen bond interactions. They can be derived from the potential function $U(\mathbf{R})$ as $\mathbf{F} = -\boldsymbol{\nabla}_{\mathbf{R}} U$.

To reach the appropriate time scale of protein internal and diffusional motion, we resort to coarse graining, where the intra- and inter-molecular mechanical forces are represented via the OPEP force field [6]. OPEP uses an off-lattice representation of a protein with five particles representing the peptidic backbone including the hydrogen and oxygen atom, and generally one particle for the side chain. Each side chain is defined by the position with respect to the amide nitrogen and carbon atoms [16]. OPEP considers protein-solvent and electrostatic interactions at an implicit level, nevertheless it provides the correct secondary and tertiary structure of proteins and is a particularly appropriate tool to study the dynamics of long biological processes and the thermodynamics of molecular assemblies.

Hydrodynamic interactions are due to the motion of single atoms that create a velocity field in the surrounding environment. The flow field acts on other particles, therefore generating effective interactions. In the force decomposition framework, the induced flow field is split as the sum of short and long wavelength components, $\mathbf{u}^{fast} + \mathbf{u}^{slow}$. The particles experience a drag force $-\gamma(\mathbf{V} - \mathbf{u}^{fast} - \mathbf{u}^{slow})$, where γ is a coupling coefficient, plus an uncorrelated stochastic noise of zero mean and variance proportional to the temperature T. We associate the intramolecular solvent-mediated interactions to the short-ranged hydrodynamic modes, with particles following an overdamped dynamics (with negligible inertia of the particles as compared to frictional forces). Thus, their positions are updated in a timestep h from the action of intramolecular hydrodynamics and mechanical forces as:

$$\mathbf{R}_h = \mathbf{R}_0 + \frac{h}{\gamma}\mathbf{H}_0 \left[\frac{\mathbf{F}_0}{\mathbf{M}} + \gamma \mathbf{u}_0^{slow}\right] + \boldsymbol{\chi}_t \qquad (1)$$

where $\boldsymbol{\chi}_t$ is a white noise with zero mean, $\langle \boldsymbol{\chi}_t \rangle = 0$, and covariance $\langle \boldsymbol{\chi}_t \boldsymbol{\chi}_{t'} \rangle = 2k_B T \mathbf{H} \gamma \delta(t - t')$. Hydrodynamic interactions are encoded by the $3N_m \times 3N_m$ Rotne-Prager-Yamakawa tensor [17]:

$$\mathbf{H}_{ij} = \frac{3}{4\zeta_{ij}} \begin{cases} \left[\left(1 + \frac{2}{3\zeta_{ij}^2}\right)\mathbf{I} + \left(1 - \frac{2}{\zeta_{ij}^2}\right)\hat{\mathbf{R}}_{ij}\hat{\mathbf{R}}_{ij}\right] & \text{if } \zeta_{ij} \geq 2 \\ \frac{\zeta_{ij}}{2}\left[\left(\frac{8}{3} - \frac{3\zeta_{ij}}{4}\right)\mathbf{I} + \frac{\zeta_{ij}}{4}\hat{\mathbf{R}}_{ij}\hat{\mathbf{R}}_{ij}\right] & \text{if } \zeta_{ij} < 2 \end{cases} \qquad (2)$$

where $\zeta_{ij} = \frac{R_{ij}}{a}$ and a is the effective particle radius, being typically smaller than the mesh spacing Δx. It should be noted that in the Brownian update (1) the total mechanical forces, stemming from intra and inter-molecular interactions, are both present.

In order to resolve the long-range hydrodynamic interactions, we determine explicitely the solvent dynamics and obtain the flow field $\mathbf{u}^{slow}$. To this aim, the scheme requires computing the particle velocities at the end of the Brownian update (1). In the overdamped picture, the velocities are irrelevant variables that are always relaxed to the equilib-

rium value given by the thermal velocity, we thus estimate the velocity as $\mathbf{V}_h = \frac{1}{\sqrt{2h/\gamma}}(\mathbf{R}_h - \mathbf{R}_0)$, whose average over noise provides the wanted thermal velocity, as derived from a formal multiscale analysis [18].

In order to determine the flow pattern at large length scales, the Lattice Boltzmann (LB) method represents a numerically robust framework to solve the Navier-Stokes equations in generic flow conditions [8]. The method is based on the evolution of the singlet distribution $f_p(\mathbf{x}, t)$, representing the probability of finding at mesh location $\mathbf{x}$ and at time t a "fluid particle" traveling with discrete speed $\mathbf{c}_p$. "Fluid particles" represent the collective motion of a group of physical particles (often referred to as populations). The mesh is three-dimensional and cartesian, a crucial advantage in terms of analytical treament, mesh management and parallel partitioning. The mesh has spacing Δx and is equipped with 19-speed vectors $\mathbf{c}_p$ (i.e., the D3Q19 scheme in LB language) connecting each mesh point to its first and second neighbors. In a timestep Δt, with $\Delta t \gg h$, the fluid populations are updated via the explicit scheme:

$$\begin{aligned} f_b(\mathbf{x} + \mathbf{c}_p\Delta t, t + \Delta t) &= \omega\Delta t f_p^{eq}(\mathbf{x}, t) + (1 - \omega\Delta t) f_p(\mathbf{x}, t) \\ &\quad + \Delta f_p(\mathbf{x}, t) \end{aligned} \tag{3}$$

The essence of the LB algorithm is to compute the RHS of Eq. (3) as a completely space-time local quantity and propagate fluid information non-locally by exchanging populations with the neighboring mesh nodes. The first term contains the collisional relaxation to local equilibria, given by the Maxwell-Boltzmann distribution:

$$f_p^{eq} = w_p\rho\left[1 + \frac{\mathbf{u}^{slow}\cdot\mathbf{c}_p}{c_s^2} + \frac{\mathbf{u}^{slow}\mathbf{u}^{slow} : (\mathbf{c}_p\mathbf{c}_p - c_s^2\mathbf{I})}{2c_s^4}\right] \tag{4}$$

where w_p are a set of weights, $\rho = \sum_p f_p$ and $\mathbf{u}^{slow} = \sum_p \mathbf{c}_p f_p/\rho$ are the fluid density and velocity, respectively.

The last term represents the effect on the fluid of the suspended particles $\Delta f_p = -w_p\Delta t\left[\frac{\mathbf{G}\cdot\mathbf{c}_p}{c_s^2} + \frac{\mathbf{G}\mathbf{u}^{slow}:(\mathbf{c}_p\mathbf{c}_p - c_s^2\mathbf{I})}{c_s^4}\right]$ via the local force $G(\mathbf{x}, t)$. In addition, $c_s = \Delta x/\sqrt{3}\Delta t$ is the fluid sound speed which is related to the space-time discretization parameters only.

The hydrodynamic fluid-particle coupling is given by a kernel constructed according to the transfer function $\tilde{\delta}(\mathbf{R})$ centered on the i-th particle [14], having compact support of size $(2\Delta x)^3$, with a corresponding hydrodynamic radius equal to $0.1\times\Delta x$. By employing a sub-mesh resolution of the particle hydrodynamic radius, with a consequent cutoff on the short wavelength of the fluid flow pattern, the separate handling of intra and inter-molecular hydrodynamic modes prevents counting these interactions twice.

Such a strategy represents a good tradeoff in terms of performances between the implicit and explicit resolution of hydrodynamic interactions. Explicitly resolving the particle-scale hydrodynamics would require a factor 10^3 more mesh points for the LB solver.

The fluid-particle coupling is based on computing the following convolution over the mesh points and for each configuration of the N suspended particles:

$$\tilde{\mathbf{u}}_i = \sum_{\mathbf{x}} \mathbf{u}^{slow}(\mathbf{x})\tilde{\delta}(\mathbf{x} - \mathbf{R}_i) \tag{5}$$

for $i = 1, ..., N$. On the fluid side, the particle-induced perturbation is encoded by the body force $\mathbf{G}(\mathbf{x}) = \sum_{i=1}^{N}\gamma(\mathbf{V}_i - \tilde{\mathbf{u}}_i)$, where $\mathbf{F}_i$ is the drag force acting on the particles and constructed according to Eq. 5.

For the particle long-wavelength dynamics, the explicit drag forces acting on the particles are encoded by the equations of motion:

$$\begin{aligned} \dot{\mathbf{R}}_i &= \mathbf{V}_i \\ \dot{\mathbf{V}}_i &= -\gamma\left(\mathbf{V}_i - \tilde{\mathbf{u}}_i\right) \end{aligned} \tag{6}$$

that are integrated in time over the timestep Δt, that is, concurrently with the LB update. Such dynamics are independent of the particle mass and consistent with the overall methodology.

The multiscale features of protein motion adapt to the computational resources in the parallel environment. While in principle the calculation of hydrodynamic interactions grows as the cube of the global system size, the evolution of fluid and particles provides a strategic algorithmic advantage. The LB method features a computational cost which scales linearly with the number of mesh points M for a fixed macromolecular number density K/M. By employing the link-cell method to compute the mechanical forces among particles, the computation also has $O(K \times N_m)$ complexity. Since the solvent-mediated particle-particle interactions are localized and explicit, the fluid-particle coupling scales linearly with the number of mesh elements and atoms.

The convolutions encoded by Eq.(5) represent a time-consuming section due to the overhead arising from the $O(10)$ number of mesh points enclosed in the particle support and scattered accesses to memory. Finally, the calculation of the hydrodynamic matrix (2) requires the calculation of the correlated noise which is done via the Cholesky decomposition method for each protein separately, with a computational cost scaling as $O(N_m^3)$.

3. CODE FEATURES

The *MUPHY* code is written in Fortran 90 and CUDA C languages, and employs MPI for communications [7]. For a flexible and efficient handling of the mesh in complex geometries, *MUPHY* makes use of an indirect addressing scheme that minimizes the memory requirements and, at the same time, facilitates the achievement of a good load balancing and employs communication pattern most suitable to the platform in use.

MUPHY was originally developed for the IBM BlueGene (L and P) architecture, where we achieved an excellent scalability up to the configuration of 294,912 cores [20]. In that case, the obtained total performance was in the range of tens of TeraFlops. There was limited room for further improvements since our code can not exploit the SIMD-like operations of the PowerPC architecture (those operations require stride-one access whereas both the LB and the MD components of *MUPHY* have a "scattered" data access pattern).

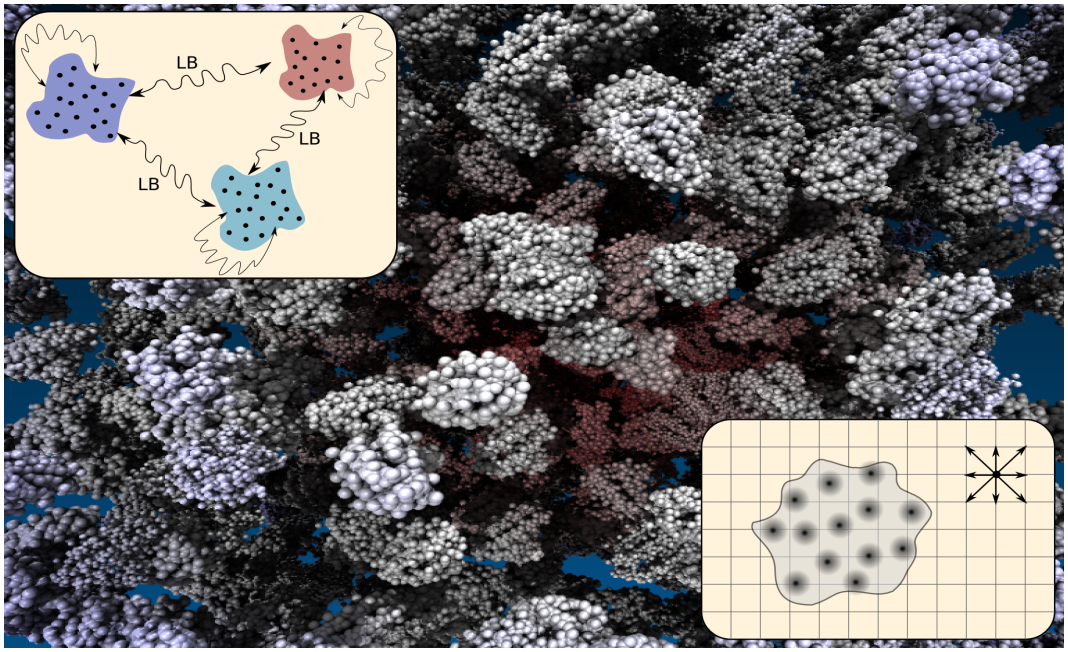

Figure 2: Detail of the protein crowding environment in a bulk simulation at 40% volume concentration. The color of the macromolecules are blue/red shaded according to their distance from the center of the box. The upper left inset sketches the intramolecular hydrodynamic interactions, as handled by the Brownian Dynamics algorithm, together with the intermolecular hydrodynamic interactions, taken care by the Lattice Boltzmann method. The lower right inset schematically shows the protein particles and associated hydrodynamic volumes, depicted as clouds, embedded in the coarse LB mesh. The arrows represent the fluid populations exchanged between neighboring mesh nodes during the LB algorithm.

The steadily increasing computing power of modern GPUs motivated the development of a version of the code targeted to clusters of GPUs [21]. With previous generation GPUs on a different problem (i.e., hemodynamics) *MUPHY* reached performance close to 1 Petaflops[19].

The multi-GPU code resorts to parallelism at two levels: intra and inter-GPU. Unlike codes developed for clusters of traditional multi-core systems that can be implemented either with a hybrid (e.g., OpenMP+MPI) or a simple distributed memory layout (counting on the availability of highly efficient MPI libraries for shared memory systems), for a multi-GPU platform a hybrid paradigm is the only choice.

This is not the only difference that needs to be taken into account: on a traditional multi-core platform the parallelism is limited to a few threads, at most tens of threads on high-end systems. On a GPU hundreds of threads are required to keep the hardware busy, so that a much finer-grained parallelization is preferred. Finally, albeit the combination of latest generation Nvidia GPUs and CUDA drivers offers the chance to copy data from/to the global (i.e., main) memory of one GPU to/from the global memory of another GPU on the same node, GPUs can not, in general, exchange data with each other without using the CPU.

The passage of data through the CPU introduces an overhead in the communication amongst the GPUs but, by using the CUDA concepts of stream and asynchronous memory copies, it is possible to overlap data transfers between GPU and CPU memory with the execution of kernels (functions in the CUDA jargon) on the GPU. Moreover, the execution of functions running on the CPU (like MPI primitives) may be concurrent with the execution of kernels on the GPU and data transfers. As a result, with an organization of the communication/computation pattern like the one shown in Fig. 3, the CPU is treated as a MPI co-processor of the GPU.

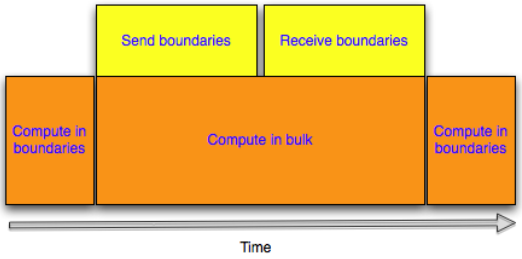

Figure 3: Communication/Computation overlap by using the CPU as a MPI co-processor of the GPU. Orange blocks are executed on the GPU whereas yellow blocks are executed on the CPU.

3.1 Communication pattern

In *MUPHY*, the communication pattern is set up at runtime. For to the LB mesh, each task determines the neighboring tasks owning the LB mesh points that need to be accessed during the simulation, regarding the non-local streaming of populations in the LB algorithm. For the particle dy-

namics, each task handles the migration of particles, the calculation of inter-domain mechanical forces and the exchange of hydrodynamics information. During this pre-processing step, we employ mainly MPI collective communication primitives. In the rest of the execution, most communications are point-to-point and make use of the following scheme: the receive operations are always posted in advance by using corresponding non-blocking MPI primitives, then the send operations are issued. Finally, each task waits for the completion of its receive operations, by using the MPI wait primitives.

For the diagnostics of physical quantities (the fluid mass and momentum, the particle energies, etc.) MPI collective (reduction) primitives are employed.

3.1.1 Particles virtual transfer

As already mentioned, the calculation of intramolecular hydrodynamics requires the execution of matrix factorizations. We could carry them out in a distributed fashion taking into account the domain decomposition employed for the fluid solver. However, such strategy is not optimal since it would be very difficult, if not impossible, to maintain a global balance of the computational load as the particles are not equally distributed in the subdomains during the simulation.

To overcome this issue, we introduced a *protein virtual transfer* algorithm, as outlined in the following. In the beginning of the simulation each protein is assigned to a single computational task such that the initial position in space of all the particles that form a protein falls within the limits of the subdomain managed by that task. Obviously, as the simulation proceeds, particles may migrate from one domain to another and particle coordinates, momenta and identities are re-allocated among tasks. However, when all the coordinates of the particles of a single protein are required to build the matrix present in Eq. 2 we *virtually* transfer *all* the particles of a protein to the task that *originally* had all the particles of that protein.

Our strategy, depicted in Fig. 4, works as follows. By using a collective communication operation, each task knows the other tasks owning particles of *its* protein (the protein belonging entirely to its subdomain in the beginning of the simulation). Each task posts a set of receive operations to collect the coordinates of those particles and, in turn, sends to other tasks the coordinates of particles belonging to proteins owned by those tasks. Once all the coordinates have been exchanged, the matrix of the coordinates (that is now completely local) is built and all tasks execute the required Cholesky decomposition in parallel without exchanging any further data. Once the decomposition is completed, the result is sent back to the tasks owning the original particles.

In this way, a perfect load balancing is achieved since each task operates on a single protein. In the current numerical simulations, the particles of a single protein are typically distributed over a small number of subdomains and the number of communications is limited, although each communication may exchange a significant fraction of the particles that constitute a protein. For the system under study, the communication/computation ratio is such that, in the end, the strategy results in a sizeable advantage in terms of global efficiency.

3.1.2 Fluid-particle coupling

An important component of our multiscale application is the fluid-particle coupling. Each suspended particle experiences hydrodynamic forces arising from the fluid macroscopic velocity, smeared over a domain made of $O(10)$ mesh points. Analogously, mesh points experience a bidirectionaly momentum transfer arising from the embedded particles.

These non-local operations require multiple communication steps such that each task owning a given particle exchanges the hydrodynamic information with the surrounding domains. The amount of momentum imparted from a particle to the surrouding fluid is transferred locally to the fluid if the particle does not overlap with the domain frontier. In the case of overlap, the portion of the particle that falls on a different domain is communicated to the domain neighbor in order to compute the fraction of momentum exchanged. For the momentum imparted from the fluid to the particle, the operation is reversed, but requires trasmitting a halo of fluid information to the task owning the particle. The latter will later perform a reduction operation over all tasks that carry the hydrodynamic information related to the particle, in order to determine the total drag force acting on the particle center of mass.

4. RESULTS

The runs whose performances are hereafter reported have been performed on the Titan system at the Oak Ridge National Laboratory (ORNL). Titan is a Cray KX7 with a hybrid architecture, one that utilizes both conventional 16-core AMD Interlagos CPUs and NVIDIA Tesla K20X GPUs. The system has 18688 nodes interconnected by the Cray Gemini network. Each node has 32GB of system memory and one GPU with 6GB of memory.

The simulation runs in single floating point precision with only few reduction operations carried out in double precision. We used the CUDA C compiler version 5.0, PGI Fortran compiler version 12.1 and Cray MPICH version 5.6.3. For the GPU-accelerated Cholesky decomposition we used the Magma library version 1.3 [22] and a custom routine.

Each GPU thread is responsible for the update, in the LB phase, of a constant number of mesh nodes (equal to 32). For the MD phase, interactions among the particles are processed on a per-particle basis. In this case, the grid of threads is directly mapped onto the arrays of particles. Threads are assigned to particles according to a global id and the search for interacting pairs proceeds for each thread in an independent fashion. Each thread scans the cell neighbors and, for each interacting pair, it computes the contribution to the total force.

Most of the performance results are presented for the largest test case. This model has $18,000$ Rat1 yeast proteins in a 3D box having dimension equal to $3,000 \times 3,000 \times 2,000$ mesh points. The whole mesh is made of 18 billion nodes and each protein has $4,013$ particles so the total number of particles is about 72 millions. *MUPHY* is able to simulate hundreds of million particles as first demonstrated in [19].

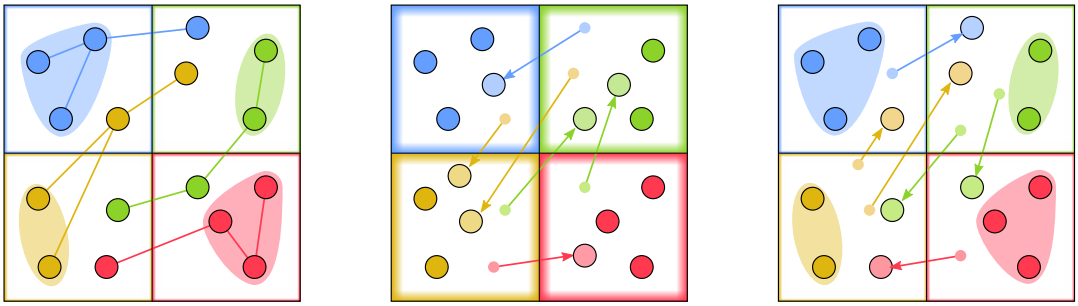

Figure 4: Protein virtual transfer in a simplified 2D case. Prior to the transfer there are particles belonging to different proteins in each subdomain. To balance the load and to carry out the matrix decomposition locally, the coordinates of the particles are "virtually" transferred to the task that originally owned the protein (central panel). In the figure the owner task is the one that has the highest number of particles of a protein. After the calculation of matrix decomposition, the results are returned to the task that currently manages the particles sent in the first step. The color of the particles is used to identify those belonging to the same protein.

4.1 Flops Count and Profiling

The Nvidia GPUs used are not equipped with counters to directly measure the number of floating points operations, although the CUDA profiler does collect useful information about kernels' (i.e. GPU functions) execution, such as the number of shared memory bank conflicts or the actual register file usage. Therefore, in order to measure the performance according to the floating point operations per second metrics, we rely on:

i) counting the number of floating point operations (Flops) by means of indirect methods;

ii) cross-checking the resulting numbers to assess their reliability;

iii) estimating the number of Flops by using the high-resolution timers provided by the GPU profiler.

The first method of counting the operations is a simple estimate based on the knowledge of the LB update and particle dynamics algorithms used in *MUPHY*. The second is a direct measure of the number of floating point operations obtained with the CPU version of *MUPHY* by using both the PAPI [23] and the oprofile [24] tools. The two methods provided results whose difference is below 10%.

For our estimates we used the smallest of the two numbers (that is, not surprisingly, the number provided by the direct measure since the optimizer can eliminate some operations during the common sub-expression reduction phase).

The most computationally intense part of the simulation is the intramolecular hydrodynamics, making use of the Cholesky decomposition. For that part of the calculation, the number of floating point operations is given by the following formula:

$$M_N + A_N$$

where

$$M_N = N \times \left[\left(\frac{1}{6} \times N + 0.5 \right) \times N + \frac{1}{3} \right]$$

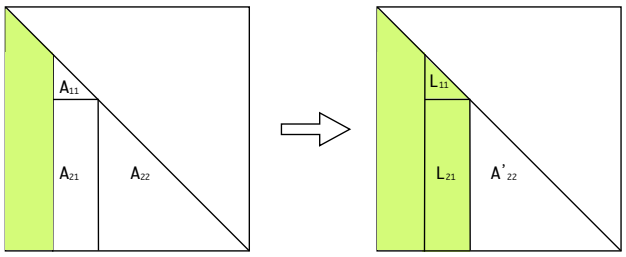

Figure 5: Right-looking blocked Cholesky decomposition.

is the number of floating point *multiply* operations and

$$A_N = N \times \left[\left(\frac{1}{6} \times N \right) \times N - \frac{1}{6} \right]$$

is the number of floating point *add* operations for a matrix of size N. The *protein virtual mechanism* allows the computation of T distinct Cholesky decompositions, where T is the total number of proteins.

In the initial version of the code we used the SPOTRF function from MAGMA. For our problem size it performs a right-looking blocked Cholesky decomposition, as depicted in Figure 5, where the following steps are performed:

1. Compute the Cholesky factorization of the diagonal block A_{11}:
$$A_{11} \leftarrow L_{11}^T L_{11}$$
2. Update the columns below the diagonal block:
$$L_{21} \leftarrow A_{21} (L_{11}^T)^{-1}$$
3. Update the trailing matrix A_{22}:
$$A_{22}^{'} \leftarrow A_{22} - L_{21} L_{21}^T$$

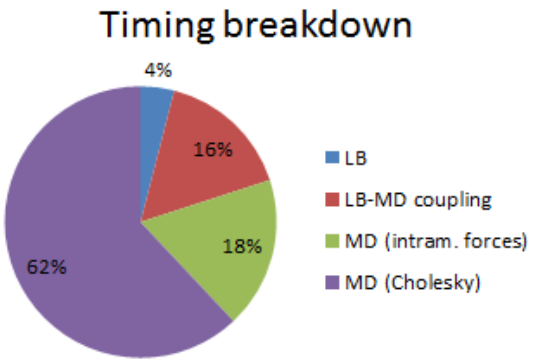

Figure 6: Breakdown of the main MUPHY components.

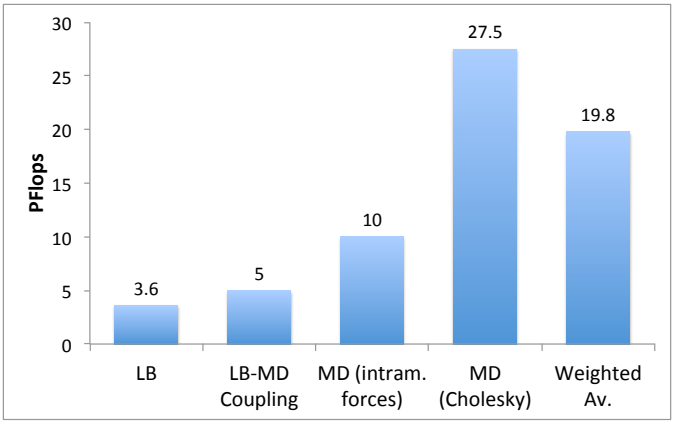

Figure 7: Relative performance. MD intramolecular forces stand for bonding and non-bonding mechanical forces, whereas the intermolecular mechanical components are negligible [16].

MAGMA performs the factorization of the panel (step 1) on the CPU and then propagates the update (steps 2 and 3, the dominant part of the computation) on the GPU. To further improve performance, we utilized a GPU implementation of SPOTRF provided by the CUDA Library team that executes the panel factorization on the GPU. We performed a parameter search for the blocking size Nb and found out that with $Nb = 3072$, we were able to achieve 1524 GFlops versus 1490 GFlops from MAGMA, as shown in Table 1.

Table 1: Gflops and time for the Cholesky decomposition of a matrix of dimension $N = 4,013 \times 3 = 12039$ for MAGMA and the custom blocked Cholesky.

Library	N	GFlops	Time
MAGMA	12039	1490	0.391s
Custom code	12039	1524	0.382s

Although the gain is just 34 Gflops, it translates to additional 0.6 PFlops in performance since there are $18,000$ GPUs executing the code in parallel. We obtained an aggregate performance of more than ***27.4*** Petaflops by using $18,000$ tasks.

In the reported timings we included the overhead due to the communication required for sending the particles to the owner task and receiving them back after the decomposition. For the Lattice Boltmann part, the raw performance of a single GPU is significantly lower due to the features of the LB method. Performance in this phase is notoriously limited by the available memory bandwidth. Nevertheless, we measured about 200 Gigaflops on a single GPU. The weak scalability of the LB part is very good, as we already reported in a previous work [19] that simulated a different biophysical problem. For the largest test-case we obtained an aggregate performance for the LB part that is close to **3.6** Petaflops on $18,000$ GPUs. Figures 6 and 7 show a timing breakdown of the main components of *MUPHY* for the $3,000 \times 3,000 \times 2,000$ testcase on $18,000$ GPUs.

Table 2: Performance of MPI routines relative to the testcase with $17,576$ tasks.

Routine	Avg Bytes	Avg Time	Bandwidth
Send (LB)	200,000	330 μsec	600 MB/sec
Waitall (LB)	N/A	750 μsec	N/A
Send (MD)	80,000	90 μsec	890 MB/sec
Waitall (MD)	N/A	3 μsec	N/A
Alltoall (MD)	4	3,670 μsec	N/A

The aggregate sustained performance (measured as weighted average of the main components) is very close to **20** Pflops.

4.2 Communication performance

Table 2 reports some information about the MPI performance obtained by using the mpiP profiler [25]. The average bandwidth for the (blocking) MPI_send operations carried out during the *protein virtual transfer* phase (marked as MD in the table) corresponds to roughly 900 MB/sec and is satisfactory in view of the highly non-trivial communication pattern. The Alltoall primitive used for the exchange of the information required by the *protein virtual transfer* mechanism takes a quite long time with respect to the limited amount of exchanged information. This is actually due to the implicit task synchronization rather than to the primitive itself.

We ran a small test by introducing a Barrier just before the Alltoall and in that situation most of the time is taken by the Barrier. The performance of the Send and Waitall operations during the *streaming* phase of the Lattice Boltzmann are, for the same reason, lower with respect to the corresponding operations carried out during the *protein virtual transfer*. The former are not synchronized whereas the latter are implicitly synchronized by the Alltoall operation that precedes them.

4.3 Scalability

The study of proteins in crowding conditions is a typical example of problems that require very large simulations in

Table 3: Time *per* step for systems of different size. The size of the fluid domain *per* task is constant and equal to 100^3.

Total Domain Size	# of proteins (& tasks)	Time / step
600^3	216	6.85 sec
700^3	343	6.7 sec
$1,000^3$	1,000	6.8 sec
$2,600^3$	17,576	6.77 sec
$3,000 \times 3,000 \times 2,000$	18,000	6.44 sec

both space (size of the system) and time. Before running the largest testcase ($3,000 \times 3,000 \times 2,000$ with $18,000$ proteins) whose performance was described earlier, we ran some (relatively) small test cases and reported the results in Table 3.

It is apparent that the code maintains a perfect weak scalability when the size of the system under study increases. Due to the requirements of the application and the limited size of the memory available on each GPU (6 GB), measuring the strong scaling performances is basically meaningless. A system with 1000 Rat1 yeast proteins simply does not fit in the memory available with, say, 343 GPUs. Each protein, including all data (for the representation of physical quantities, fluid-particles coupling, communication, *etc.*), requires slightly less than 5 GB. The Lattice Boltzmann part for a subdomain having 100^3 mesh points, requires about 300 MB. On the other hand, using, for instance, 1000 GPUs to simulate fewer proteins is a waste of resources. As mentioned earlier, it is much more efficient to execute the most computing intensive part of the simulation (i.e., the Cholesky decomposition) by handling an entire protein on a single GPU.

5. PROTEIN DIFFUSIVITY

The simulation data gathered for an exhaustive ensemble of biomolecules provides a wealth of microscopic information, a precious database to understanding the diffusional features and structural modifications at volume fractions spanning the range of values observed in different cellular compartments [1]. In this section, we report the diffusional features as an example of simulation capabilities obtained on the large dataset as accessed via the Titan platform.

We analyzed the diffusional features of a suspension made of $17,576$ Rat1 proteins in solution at room temperature. The Rat1 yeast protein from the S. Pombe, is a biomolecule composed by 666 aminoacids that functions primarily in the nucleus. It also has an important role in transcription.

In the OPEP representation, 666 aminoacids correspond to $4,013$ particles. The simulations were performed at 10, 20, 30 and 40 % volume fractions. A preliminary simulation of a single protein at virtually infinite dilution was used to estimate the protein volume via the Voronoi construction algorithm. Starting from the Brookhaven Protein database structure, an initial single protein was relaxed according to a steepest descent procedure and then simulated over $10\,ns$ with standard Molecular Dynamics. Finally the protein was replicated and each protein placed on a cubic lattice, resulting in $26 \times 26 \times 26$ proteins in the largest configuration at hand. In this configuration, a relaxation cycle has been performed before starting the complete production run. Such relaxation did not produce substantial displacement of the proteins' centers of mass.

In the simulation protocol, the Brownian updates were performed with a 10 step subcycle of the particle intramolecular hydrodynamics and mechanical interactions for each fluid and long wavelength particle update, guaranteeing an optimized and stable procedure. Analysis of the macromolecular trajectories revealed that during their erratic motion, the encounters among proteins presents a high degree of heterogeneity, with several non-specific and specific interactions. The data for the diffusion coefficient were gathered over $10\,ns$ and the translational diffusion coefficient was determined via the Einstein relation by monitoring the displacements of the protein centers of mass. The diffusion coefficients are shown in Fig. 8, together with the experimental data obtained by quasielastic neutron scattering experiments for the bovin serum albumin protein [4].

The computed diffusion coefficients provide similar, although systematically larger, values than the experimental ones, probably due to the larger temporal scale accessed by the simulation as compared to the time window pertinent to quasielastic neutron scattering spectra ($3.5\,ns < \tau_{expt} < 5\,ns$). This temporal scale exceeds the hydrodynamic timescale arising from the propagation of vorticity over the protein linear size ($\sim 100\,ps$) and slows down the protein self-diffusion.

The rapid drop of the diffusion coefficient for volume fraction comprised between 10 and 30 % appears to signal the onset of caging mechanisms while at larger volume fraction the diffusivity approaches, but does not yet reach, a jamming transition. Further analysis will provide more quantitative information on such transition, together with an extensive analysis of the protein structural fluctuations and dynamical features.

6. CONCLUSIONS

We have presented results from large-scale simulations of protein suspensions in crowding conditions on a world-class cluster of GPUs, enabled by recent enhancements of the *MUPHY* code, our general tool for research in multiscale biofluidics.

For the largest test case, the computational environment involves up to 18 billion fluid nodes coupled with the concurrent motion of thousands of proteins. For that system, we have achieved close to 20 Petaflops of sustained performance on $18,000$ GPUs of the ORNL Titan supercomputer, with a peak of 27.5 Petaflops for the parallel Cholesky decomposition component of the simulation.

The above accomplishment results from the combination of several features, both from high-performance computing technology and from physical/computational modeling. In particular: *i*) the usage of a genuinely multiscale methodology to solve self-consistently intra and inter-molecular hydrodynamics for a complex biological system, optimizing the number of computational resources involved; *ii*) the *virtual* transfer technique that secures a balanced and localized

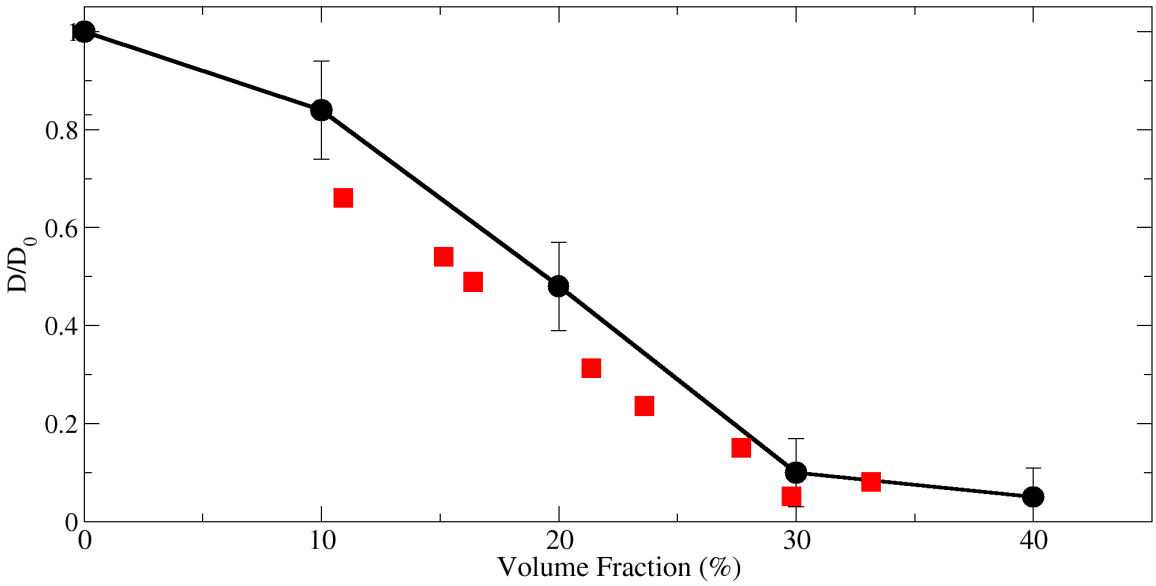

Figure 8: Translational diffusion coefficient at various volume fractions (black circles), compared with experimental data for bovin serum albumin [4] (red squares). The diffusion coefficient has been normalized by the value at virtually zero volume fraction. The solid line is a guide to the eye.

workload during the calculation of intramolecular hydrodynamics; and *iii*) the usage of CPUs as MPI-coprocessors of the GPUs that transformed an intrisic limitation of the GPU (the lack of support for direct communication) into an advantage.

The present achievement marks a dramatic advance in the predictive capabilities of computer simulation to address complex biological systems. The accurate description of the concurrent motion of fluid-flow and suspended biomolecules opens the way to a qualitatively new understanding of the complex bio-chemical-fluid-mechanical interactions laying at the basis of cell function. The scientific and societal impact of such a perspective cannot be understated.

7. ACKNOWLEDGMENTS

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. We wish to thank Jack Wells and Don Maxwell of Oak Ridge National Laboratory for their support. Computing time at the CINECA supercomputing center on both the Blue Gene/Q Fermi system (ISCRA grant FLEXPROT) and the PLX DataPlex GPU cluster (PRACE grant DEC08_LBglaSS) is also kindly acknowledged.

We also wish to thank Sauro Succi, Umberto Marini Bettolo Marconi, Fabio Sterpone, Carlo Pierleoni, Alessandro Fatica and Philippe Derreumaux for relevant discussions and Gregory Ruetsch and Everett Phillips for help with the paper and the runs. A special thank goes to Philippe Vandermersch for providing the SPOTRF on GPU used as a building block for the blocked Cholesky factorization in the latest run on the 18,000 GPUs of the Titan system.

8. REFERENCES

[1] R.J. Ellis, A. P. Minton, Nature, 425, 27 (2003).
[2] Y. Wang, C. Li, G.J. Pielak, J. Am. Chem. Soc., 132, 9392-9397 (2010).
[3] R. Harada, Y. Sugita, M. Feig, J. Am. Chem. Soc. 134, 4842-4849 (2012).
[4] F. Roosen-Runge, M. Hennig, F. Zhang, R. M.J. Jacobs, M. Sztucki, H. Schober, T. Seydel, F. Schreiber, Proc. Natl . Acad. Sci USA, 108, 11815-11820 (2011).
[5] A. Korkut, W.A. Hendrickson, Proc. Natl. Acad. Sci. USA, 106 15667-15672 (2009). S.J. Marrink, H.J. Risselada, S. Yefimov, D.P. Tieleman, A.H. de Vries, J. Phys. Chem. B. 111 7812-7824 (2007). M. Neri, C. Anselmi, M. Cascella, A. Maritan, P. Carloni, Phys. REv. Lett. 95, 218102 (2005).
[6] P. Derreumaux, N. Mousseau, J. Chem. Phys. 126, 025101 (2007).
[7] M. Bernaschi, S. Melchionna, S. Succi et al., Comp. Phys. Comm., 180, 1495 (2009)
[8] R. Benzi, S. Succi and M. Vergassola, Phys. Rep., 222, 145 (1992).
[9] D. Moroni B. Rotenberg, J.-P. Hansen, S. Succi, S. Melchionna, Phys. Rev. E, 73, 066707 (2006).
[10] S. Melchionna, Macromol. Theory Sim., DOI: 20,

548-558 (2011).
[11] S. Melchionna, J. Comput. Phys. 230, 3966 (2011).
[12] S. Melchionna, M. Bernaschi, S. Succi et al, Comp. Phys. Comm., 181, 462 (2010).
[13] U. Marini Bettolo Marconi, S. Melchionna, Langmuir, 28, 13727-13740 (2012).
[14] M. Bernaschi, M. Bisson, M. Fatica, S. Melchionna, S. Succi, Comp. Phys. Comm. 184, 329-341 (2013).
[15] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, USA, (2001)
[16] The OPEP force field is expressed as a sum of local, nonbonded, and hydrogen-bond terms. The short-range potential is of the form $U_{sr} = \sum_{bonds} K_b(R - R_{eq})^2 + \sum_{angle} K_\alpha(\alpha - \alpha_{eq})^2 + \sum_{dihedrals} K_\Omega(\Omega - \Omega_{eq})^2 + \sum_\phi K_\phi(\phi - \phi_{eq})^2$, with bond, angular, improper dihedral harmonic and Ramachandran restraining potentials (where R, α, Ω, ϕ indicate the corresponding bond distances, angles, dihedrals and Ramachandran angles), respectively. The nonbonded interactions are of the form, $U_{nb} = \sum_{1-4} U_{LJ}(R) + \sum_{pairs} U_{LJ}(R)$, involving 1-4 topological neighbors and generic atom pairs, where $U_{LJ} = \epsilon \left[\left(\frac{\sigma}{R}\right)^{12} - \left(\frac{\sigma}{R}\right)^{6} \right]$ is a Lennard-Jones potential. The interaction is complemented by a set of hydrogen bond and cooperative potential terms, involving pairs and quadruplets of labelled particles.
[17] J. Rotne, S. Prager, J. Chem. Phys. 50, 4831 (1969).
[18] U. Marini Bettolo Marconi, S. Melchionna, J. Chem. Phys. 126, 184109 (2007).
[19] M. Bernaschi, M. Bisson, T. Endo, M. Fatica, S. Matsuoka, S. Melchionna, S. Succi, "Petaflop biofluidics simulations on a two million-core system", Gordon Bell Finalist. Proceedings of Supercomputing 2011, Seattle (U.S.A.) (2011).
[20] A. Peters, S. Melchionna, E. Kaxiras, J. Latt, J. Sircar, M. Bernaschi, M. Bisson, S. Succi, "Multiscale simulation of cardiovascular flows on the IBM Bluegene/P: full heart-circulation system at near red-blood cell resolution", Gordon Bell Finalist. Proceedings of Supercomputing 2010, New Orleans (U.S.A), (2010).
[21] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi and E. Kaxiras, Concurrency and Computation: Practice and Experience, 22, 1 (2010).
[22] "MAGMA: Matrix Algebra on GPU and Multicore Architectures", `http://icl.cs.utk.edu/magma/index.html`
[23] K. London, J. Dongarra, S. Moore, P. Mucci, K. Seymour, T. Spencer, International Conference on Parallel and Distributed Computing Systems, Dallas, TX, August 8-10, (2001).
[24] `http://oprofile.sourceforge.net`.
[25] C. Chambreau, J. Vetter, "mpiP: Lightweight, Scalable MPI Profiling", `http://mpip.sourceforge.net/`